Java II

Final

1) A class (ClassOne) is considered to have a dependency on another class (ClassTwo) under which of the following conditions?

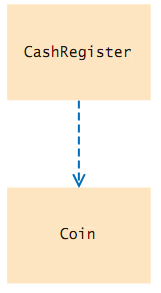
a) Each class uses objects of the other.

b) The public interfaces of both classes are cohesive.

c) ClassTwo uses objects of ClassOne.

d) ClassOne uses objects of ClassTwo.

2) Which statement correctly describes the class relationship shown in this diagram?



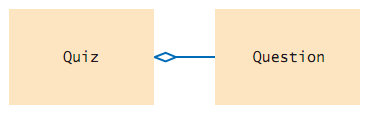
a) CashRegister class depends on Coin class

b) Coin class depends on CashRegister class

c) CashRegister class aggregates Coin class.

d) Coin class inherits from CashRegister class.

3) Which statement correctly describes the class relationship shown in this diagram?



a) Quiz class depends on Question class

b) Question class aggregates Quiz class

c) Quiz class aggregates Question class.

d) Question class inherits from Quiz class.

4) Consider the following code snippet:

public class SailBoat extends Vessel

{ . . . }

public class Catamaran extends SailBoat

{ . . . }

Which of the following statements is NOT correct?

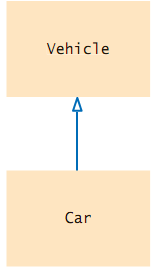
a) SailBoat inherits from Vessel.

b) Catamaran inherits from Vessel.

c) Catamaran inherits from Sailboat.

d) Catamaran depends on SailBoat.

5) Which statement correctly describes the class relationship shown in this diagram?



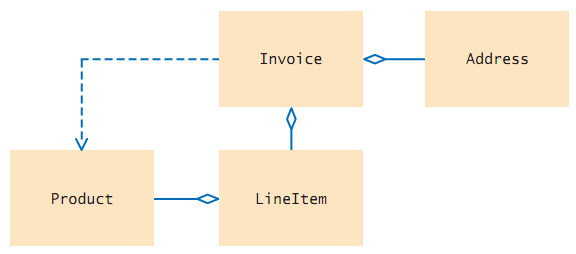
a) Vehicle class depends on Car class

b) Car class depends on Vehicle class

c) Vehicle class inherits from Car class.

d) Car class inherits from Vehicle class.

6) Given the following diagram showing class relationships:



What type of relationship is shown between Invoice and LineItem?

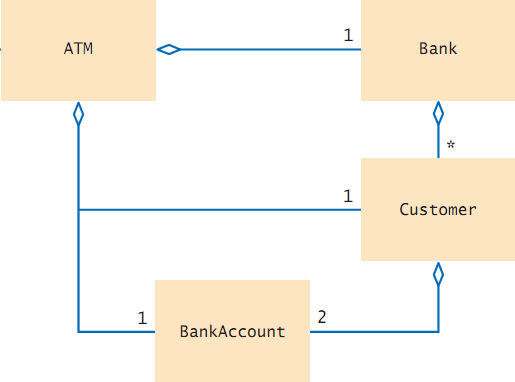
a) LineItem inherits from Invoice.

b) LineItem aggregates Invoice.

c) Invoice aggregates LineItem.

d) Invoice inherits from LineItem.

7) Given the following diagram showing class relationships:



What type of relationship is shown between BankAccount and Customer?

a) Customer aggregates BankAccount, indicating that a customer may have 2 bank accounts.

b) BankAccount aggregates Customer, indicating that 2 customers may share a bank account.

c) BankAccount depends on Customer, indicating that many customers may have 2 bank accounts.

d) Customer depends on BankAccount, indicating that a customer may have 2 bank accounts.

8) Consider the getArea method from the textbook shown below:

public int getArea()

{

if (width <= 0) { return 0; } // line #1

else if (width == 1) { return 1; } // line #2

else

{

Triangle smallerTriangle = new Triangle(width – 1); // line #3

int smallerArea = smallerTriangle.getArea(); // line #4

return smallerArea + width; // line #5

}

}

Assume the code in line #3 is changed to:

Triangle smallerTriangle = new Triangle(width);

This change would cause infinite recursion for which triangles?

a) Those with width equal to 0.

b) Those with width equal to 1.

c) Those with width greater than or equal to 2.

d) Triangles of any width.

9) Consider the following recursive code snippet:

public static int mystery(int n, int m)

{

if (n <= 0)

{

return 0;

}

if (n == 1)

{

return m;

}

return m + mystery(n - 1, m);

}

Identify the terminating condition(s) of method mystery?

a) n <= 0

b) n == 1

c) n <= 0 or n == 1

d) n > 0

10) Complete the code for the recursive method printSum shown in this code snippet, which is intended to return the sum of digits from 1 to n:

public static int printSum(int n)

{

if (n == 0)

{

return 0;

}

else

{

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

}

a) return (printSum(n - 1));

b) return (n + printSum(n + 1));

c) return (n + printSum(n - 1));

d) return (n - printSum(n - 1));

11) Insert the missing code in the following code fragment. This fragment is intended to recursively computexn, where x and n are both non-negative integers:

public int power(int x, int n)

{

if (n == 0)

{

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

else

{

return x \* power(x, n - 1);

}

}

a) return 1;

b) return x;

c) return power(x, n - 1);

d) return x \* power(x, n - 1);

12) How many recursive calls to the fib method shown below would be made from an original call to fib(4)? (Do not count the original call)

public int fib(int n)

{ // assumes n >= 0

if (n <= 1)

{

return n

}

else

{

return (fib(n - 1) + fib(n - 2));

}

}

a) 1

b) 2

c) 4

d) 8

13) Consider the method below, which implements the exponentiation operation recursively. Select the statement that should be used to complete the method, so that it handles the special case correctly.

public static double power(int base, int exponent)

{

if (exponent == 0)

{

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

else

{

reurn base \* power(base, exponent – 1);

}

}

a) return 1;

b) return base;

c) return 0;

d) return 1 \* power(base, exponent – 1);

14) Consider the code for the recursive method printSum shown in this code snippet, which is intended to return the sum of digits from 1 to n:

public static int printSum(int n)

{

if (n <= 0) // line #1

{

return 0; // line #

}

else

{

return (n + printSum(n)); //line #3

}

}

Which of the following statements is correct?

a) line #1 is incorrect, and should be changed to if (n <= 1)

b) line #3 is incorrect, and should be changed to return (printSum (n - 1));

c) line #3 is incorrect, and should be changed to return (n + printSum (n + 1));

d) line #3 is incorrect, and should be changed to return (n + printSum (n - 1));

15) Consider the sort method for selection sort shown below:

public static void sort (int[] a)

{

for (int i = 0; i < a.length – 1; i++)

{

int minPos = minimumPosition(i);

swap(minPos, i);

}

}

Suppose we modify the loop control to read int i = 1; i < a.length – 1; i++. What would be the result?

a) An exception would occur

b) The sort would not consider the last array element.

c) The sort would not consider the first array element.

d) The sort would still work correctly.

16) Consider the minimumPosition method from the SelectionSorter class. Complete the code to write a maximumPosition method that returns the index of the largest element in the range from index from to the end of the array.

private static int minimumPosition(int[] a, int from)

{

int minPos = from;

for (int i = from + 1; i < a.length; i++)

{

if (a[i] < a[minPos]) { minPos = i; }

}

return minPos;

}

private static int maximumPosition(int[] a, int from)

{

int maxPos = from;

for (int i = from + 1; i < a.length; i++)

{

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

return maxPos;

}

a) if(a[i] > a[maxPos]) { maxPos = i; }

b) if(a[i] == a[maxPos]) { maxPos = i; }

c) if(a[i] < a[maxPos]) { maxPos = i; }

d) if(a[i] <= a[maxPos]) { maxPos = i; }

17) The merge sort algorithm presented in section 14.4, which sorts an array of integers in ascending order, uses the merge method which is partially shown below. Select the condition that would be needed to complete the method so that the elements are sorted in descending order.

private static void merge(int[] first, int[] second, int[] a)

{

int iFirst = 0;

int iSecond = 0;

int j = 0;

while (iFirst < first.length && iSecond < second.length)

{

if (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

{

a[j] = first[iFirst];

iFirst++;

}

else

{

a[j] = second[iSecond];

iSecond++;

}

j++;

}

// rest of the method follows here

}

a) first[iFirst] < second[iSecond]

b) iFirst < iSecond

c) first[iFirst] > second[iSecond]

d) iFirst > iSecond

18) Given the following code snippet for searching an array:

int[] arr = {3, 8, 12, 14, 17};

int newVal = 15;

int pos = Arrays.binarySearch(arr, newVal);

What value will pos have when this code is executed?

a) 5

b) -5

c) 6

d) -6

19) Assume that bands is an ArrayList of String objects which contains a number of elements in ascending order. Select a statement to complete the code segment below, which invokes the Java library binarySearch method to search for the string "Beatles". If the list does not already contain the string, it should be inserted in an appropriate location so that the list remains sorted.

int index = Collections.binarySearch(bands, "Beatles");

if (index < 0)

{

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

a) bands.add(-1 \* index + 1, "Beatles");

b) bands.add(index + 1, "Beatles");

c) bands.add(-1 \* index, "Beatles");

d) bands.add(-1 - index, "Beatles");

20) The partial binary search method below is designed to search an array of String objects sorted in ascending order. Select the expression that would be needed to complete the method.

public static int search(String[] a, int low, int high, String item)

{

if (low <= high)

{

int mid = (low + high) / 2;

int result = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

if (result == 0)

{

return mid;

}

else if (result < 0)

{

return search(a, mid + 1, high, item);

}

else

{

return search(a, low, mid - 1, item);

}

}

return -1;

}

a) a[low].compareTo(item)

b) item.equals(a[mid])

c) item.compareTo(a[mid])

d) a[mid].compareTo(item)